Using Natural Language Policies for Privacy Control in Social Platforms*

Juri L. De Coi[1], Philipp Kärger[1], Daniel Olmedilla[2], and Sergej Zerr[1]

[1] L3S Research Center & Leibniz University of Hannover, Germany
[2] Telefónica Research & Development, Madrid, Spain

Abstract. The ability of defining privacy preferences in the current social platforms are very restricted. Typically, the user is provided with only some predefined options to select from. In this paper, we present an approach that exploits Semantic Web policies allowing users to control privacy in social web applications. Such policies are general statements that define the behavior of a system. Although Semantic Web policies gained a lot of interest in recent years and policy languages became more and more complex, suitable and easy-to-use solutions for highly dynamic social platforms are still needed. In order to allow *common* users to define policies about the data to be shared, we introduce *natural language* policies. We further present an implementation based on the policy framework Protune that allows users of a collaborative learning platform to restrict access to their learning material to collaborators by means of controlled natural language policies.

1 Introduction

Since the web became a place where people are not only consuming content but creating, publishing and sharing content, it is needed to allow people to exactly define who is allowed to access which part of the content they provide. Unfortunately, there is currently no simple way to restrict access to some content to only a set of trusted parties not previously known or to decide who is allowed to access some part of a profile [1]. In Flickr for instance, you are allowed to define people as friends or family members and, based on those assignments, it can be decided who is allowed to see what picture. Still more fine-grained access control management is lacking. Thus, if both, colleagues and close friends, are in the same group, there is no way to restrict access to a subgroup for a particular picture [1]. Other web applications for sharing data or social platforms do also allow only for similar limited access control features leading to a high potential of leaking private data [2].

Semantic Web policies have gained a lot of interest in the last years and promising approaches show their applicability to real-life applications, be they in the area of web personalization, agent and network control, or access control on the web. For this reason a number of policy languages have been defined in the last years. Nevertheless a major hindrance to widespread adoption of policy languages are their shortcomings in terms of usability: in order to be machine-understandable all of them rely on a formal syntax, which common users find unintuitive and hard to grasp. Therefore, in most such applications, policies are required to be authored by system administrators or specifically trained people.

* The authors' efforts were partly funded by the European Commission in the TENCompetence project (IST-2004-02787) (http://www.tencompetence.org).

In this paper, we present an approach that overcomes both mentioned shortcomings: the limited access control capabilities of existing collaborative systems and the complexity of current policy languages. First, we do not restrict the user to role-based access control on the basis of user groups or similar but we allow users to specify general policies providing fine-grained access control. Second, our approach does not require users to define policies in formal languages but provides an interface for natural language policies.

In the presented approach, we apply Protune and its corresponding policy language. By doing so we additionally benefit from the features of an advanced policy engine such as the generation of natural language text explaining the result of a policy evaluation (e.g., why access was denied) or the support of trust negotiation [3]. Further, we use an adapted version of the controlled natural language ACE to define policies.

The remainder of this paper is structured as follows. In the next section, we identify requirements a general privacy control solution should fulfill. In Section 3 we provide some background information on Semantic Web policies and controlled natural languages. Then, in Section 4 and 5 we detail how policies can be used to control privacy in social web applications. Section 6 introduces our prototype and Section 7 concludes the paper.

2 Motivation and Problem Statement

Preserving privacy in social platforms is a known challenge (see [4] for an information leakage on Facebook or [5] for a similar case on Flickr). Existing social platforms provide only limited means to define access control rights [6]. The problem is that privacy preferences have to be easy to define but still expressive enough to capture all possible cases and combinations of a user's wishes to define who is allowed to access what. An additional challenge is the dynamics of nowadays social networks. We identified the following requirements for a solution aiming at preserving privacy on a social platform.

Adaptive to social network dynamics. Since nowadays social networks change rapidly, any solution should be able to easily adapt itself to changes in the data of requesters.

Fine-grained. Privacy preferences can be arbitrary fine-grained. If a picture shall be shared only with a restricted set of people (maybe not even known in advance), it should be easy to express such requirement.

Extensible. Privacy preferences should be easily extensible: imagine a user uploads a new set of pictures to a social platform and she wants them to be shared with attendees of a conference. The user's current privacy preferences should be easily extensible without requiring all conference attendees to be registered at the social platform.

Natural language interface and feedback. Defining privacy preferences has to remain a simple and straightforward process. Access control decisions should be transparent and well explained to users. Similarly, the specification of privacy preferences has to protect users from collections of check boxes defining which friends is allowed to access which file or from similar complicated policy definitions.

Security mechanisms. Last, but not least, any solution must fulfill basic security and privacy requirements, such as reliability, support to authentication, delegation of rights and evidences (such as credentials and declarations), etc.

3 Background

3.1 Semantic Web Policies

Semantic Web policies are statements defining the behavior of a system. Access control policies define the conditions to be fulfilled in order to get access to a certain resource. Policies are pervasive in all web-related contexts [7] and are needed to protect any system open to the internet. They declaratively specify requirements which must be fulfilled but do not state how they have to be fulfilled, thereby providing a separation between specification and enforcement. Privacy policy frameworks such as the Platform for Privacy Preferences (P3P) or EPAL are exploited to assist users while they are browsing the web and interacting with web services. The early policy frameworks were rather decision support systems than systems for declarative behavior control [7]. Moreover, they were not expressive enough to fully address Semantic Web scenarios involving reasoning, ontology-based knowledge representation, etc. Nowadays, a number of policy languages have been developed in order to address the challenges of the Semantic Web: expressiveness, simplicity, enforceability, scalability, and analyzability [8]. Among the most prominent ones (see [14] for a broad overview of policy languages and their features) we mention KAoS [9], Rei [10], PeerTrust [11], and Protune [12]. In order to provide a well-defined semantics these policy languages are typically based on languages logical formalisms like description logics (e.g., KAoS and Rei) or logic programming (e.g., Cassandra [13], PeerTrust, and Protune). Policy engines are in charge of evaluating policies (i.e., of checking whether they are satisfied or not).

The Policy Framework Protune
In our approach we used the Protune[3] (PRovisional TrUst NEgotiation) framework for policy evaluation and enforcement. Protune [12] aims at combining distributed trust management policies with provisional-style business rules and access control-related actions. Protune features an advanced policy language for policy-driven negotiation and supports distributed credentials management and flexible policy protection mechanisms. The Protune policy framework offers a high flexibility for specifying any kind of policy, referring to external systems from within a policy and providing facilities for increasing user awareness, like automatic generation of natural language explanations of the result of a policy's evaluation.

The Protune policy language is Logic Programming-based and as such a Protune policy has much in common with a Logic Program. An example Protune policy is

$$allow(access(Requester, Resource)) \leftarrow friend(Requester), family\_picture(Resource).$$

states that a requester can access a resource if the requester is a friend and the resource is a family picture. (According to the standard Logic Programming conventions, terms starting with a capital letter refer to variables; that is, *Resource* indicated that the policy applies to every resource.) Adding the following facts $family\_picture('holiday.jpg')$. and $friend('Bob')$. will allow Bob to access the picture `holiday.jpg`. In general, a Protune policy rule has one of the following formats:

$$allow(action) \leftarrow condition_1, \ldots, condition_n.$$

[3] http://policy.L3S.uni-hannover.de/

$$condition \leftarrow condition_1, \ldots, condition_n.$$

(where $n \geq 0$), meaning that the action *action* is allowed (resp. the condition *condition* holds) if all conditions $condition_i$ hold.

A policy may require that, in order to evaluate a request, some actions are performed, that is, some of the $condition_i$ can be actions themselves. For example, rule

$$allow(action_1) \leftarrow action_2.$$

can be read as: $action_1$ *can be* executed if $action_2$ *has been* executed. In a typical scenario access to some resource is allowed only if the requester provides a valid credential (in this case, $action_2$ would be sending a credential). Notice the different semantics of the actions according to the side of the rule they appear in: the execution of actions appearing in the left-hand can be requested by some party, whereas the actions appearing in the right-hand side have to be performed in order to accomplish the request. In order to stress this semantic difference, actions in the left-side of a rule must appear inside the *allow* predicate.

The evaluation of a policy may require to deal with objects which can be modeled as sets of $(attribute, value)$ pairs. The Protune language allows to refer to such properties of an object by means of so-called *complex terms*.

$$identifier.attribute : value$$

For example, a rule stating that only credit cards whose company is VISA are accepted could look like the following.

$$accepted(CreditCard) \leftarrow CreditCard.company :' VISA'.$$

3.2 Controlled Natural Languages

Controlled natural languages (CNLs) are formal languages which have been designed in order not to look like formal languages but to be more user-friendly. As formal languages they are described by a formal grammar each well-formed sentence must comply to. Moreover, they embed disambiguation rules in order to deterministically disambiguate sentences which in full natural language have different readings, as it happens with the following standard example: *Bob looks at the girl on the hill with a telescope.* In full natural language such sentence can have at least three different meanings according to the context it appears in, namely

- *Bob looks with a telescope at the girl who is on the hill.*
- *Bob looks at the girl who is on the hill and has a telescope.*
- *Bob is on the hill, has a telescope and looks at the girl.*

Although the last reading may be considered a bit unlikely, it cannot be excluded. In order to disambiguate such sentences, CNLs usually come along with a set of built-in rules which allow to deterministically select one out of n possible readings. For instance the CNL ACE assumes that all prepositional phrases refer to the predicate, therefore in the example above the action "looking" happens "towards the girl", "on the hill" and "with a telescope", boiling down to the third one of the readings listed above.

Ambiguity resolution takes place on a purely syntactic basis, so that CNLs typically do not differentiate between "God's sake" and "John's love", being both of them built

according to the pattern `[proper name]'s [common noun]`, although in everyday's speech one probably means that it is God to be loved whereas it is John to love.

The ACE controlled natural language

In our approach we use Attempto Controlled English (ACE [15]). ACE is a controlled natural language developed at the University of Zurich with the aim to serve as specification and knowledge representation language on the Semantic Web. ACE is intended for professionals who "need to use formal notations and formal methods, but may not be familiar with them"[4]. The Attempto Parsing Engine (APE) deterministically translates ACE texts into a semantically equivalent internal format (Discourse Representation Structure—DRS) which is easier to further process in an automatic way. ACE and APE come along with a set of tools able to translate DRSs into various other languages (e.g., FOL, PQL, FLUX, RuleML) and a rule format to be used for Courteous Logic Programs [16] and stable model semantics [17]. A prototypical bidirectional translation of ACE into and from OWL DL has been developed as well.

4 Controlled natural language for easy specification of privacy policies

A major hindrance to widespread adoption of policy languages are their shortcomings in terms of usability: in order to be machine-understandable all of them rely on a formal syntax, which common users find unintuitive and hard to grasp. The use of controlled natural languages can improve usability of policy languages. This section describes how we exploit (a subset of) ACE in order to express policies and describe the mapping between ACE policies and the policies written in the Protune language. The full details of the ACE→Protune translation are available in [18]. The following ACE policies are corresponding to the Protune policies introduced in Section 3.1.

- If the requester is a friend and the resource is a family-picture then the requester can access the resource.
- If the company of a credit-card is "VISA" then the credit-card is accepted.

These examples show which features of (ACE and therefore of) the English language can be used in order to define Protune policies: common nouns (like `requester`), adjectives (like `accepted`), verbs (like `access`), genitives (like in `company` *`of`* `a credit-card`) and strings (like `"VISA"`). The following example also shows that proper names (e.g., `Bob`), adjectives in comparative (e.g., `older than`) as well as in superlative form, integers (e.g., `18`) as well as reals, prepositional phrases (e.g., `in "adult-content-folder"`), saxon genitives (e.g., `Bob's`) and relative pronouns (e.g., `everything` *`which`* `is`) are supported too.

> If the requester is older than 18 and she is Bob's friend then she can access everything which is in "adult-content-folder".

As described in Section 3.1, Protune policies conform to given patterns. It is therefore intuitive that ACE policies (meant to be translated into Protune policies) also must conform

[4] http://attempto.ifi.uzh.ch/site/

to predefined patterns. It is indeed the case since, roughly speaking (more on this in [18]), ACE policies must have one of the following formats.

- If $condition_1$ and ... and $condition_n$ then *action*.
- If $condition_1$ and ... and $condition_n$ then *condition*.

where $n \geq 0$, *condition* and $condition_i$ $(0 \leq i \leq n)$ are phrases containing the features listed above (e.g., `the requester is a friend` or `the resource is a family-picture`) and *action* is a phrase containing (beside the features listed above) the modal verb `can` (e.g., `the requester can access the resource`). As the reader can see, all policies presented in this section as well as those listed in Table 1 and 2 conform to these patterns.

After having introduced the ACE→Protune mapping in an intuitive way, we conclude this section by providing a more principled approach to the translation by listing the most remarkable mapping rules we defined.

A programmer asked to formalize the sentence *John sends Mary the credential* as a logic program would most likely come up with a rule like $send(john, mary, credential)$. Indeed in many cases translating verbs into predicate(name)s can be considered the most linear approach, which we pursued as well. However this approach only allows to produce predicates with a number of parameters up to three. In order to support predicates with a bigger number of parameters we decided to exploit ACE propositional phrases. For instance, sentence *If ... then John pays the book with the credit-card.* is translated to $'pay\#with'('John', Book, Credit\_card) \leftarrow \ldots$

The statement *John is Mary's friend* can be seen as asserting some information about entity "Mary", namely that the value of her property "friend" is "John". It should be then intuitive exploiting Protune complex terms (recall Section 3.1) to map such ACE sentence to $'Mary'.friend :' John'$.

When translating noun phrases like *a user* it must be decided if it really matters whether we are speaking about a "user" (in which case the noun phrase could be translated as $user(X)$) or not (in which case the noun phrase could be translated as a variable X). According to our experience, policy authors tend to use specific concepts even if they actually mean generic entities. For this reason we followed the second approach, according to which the sentence *If a user owns a file then the user can access the file.* is translated into the Protune rule: $allow(access(User, File)) \leftarrow own(User, File)$. If it is needed to point out that the one owning a file is a user, the sentence can be rewritten e.g., as follows: *If X is a user and X owns a file then X can access the file.* which gives the translation: $allow(access(X, File))$ $\leftarrow user(X), own(X, File)$.

In the following section we will detail how such natural language policies can be used for privacy preservation in a social platform.

5 Policy Enforcement for Preserving Privacy in Social Platforms

Using policies and the policy framework Protune for access control provides the user with a fine-grained specification of his privacy preferences in a declarative manner. So far we explained how natural language eases the specification of access control policies. In this section we explain how enforcing those policies preserves privacy in social data sharing platforms.

ACE Policy	Protune Policy
P1 If a document is protected then if the requester sends a student-id and the student-id's issuer is "uni-hannover" then the requester can access the document.	$allow(access(Requester0, Document0)) \leftarrow$ $protected(Document0),$ $send(Requester0, Student\_id0),$ $Student\_id0.issuer :' uni\_hannover'.$
P2 Every requester can access everything which is an employee-credential.	$allow(access(Requester0, Something0)) \leftarrow$ $'employee\_credential'(Something0).$

Table 1. A lecturer's policy, in CNL and its corresponding Protune translation.

ACE Policy	Protune Policy
P3 If a document is protected then if the requester sends an employee-credential C and the issuer of C is "uni-hannover" then the requester can access everything which is a student-id.	$allow(access(Requester0, Something0)) \leftarrow$ $send(Requester0, Employee\_credential0),$ $Employee\_credential0.issuer :' uni\_hannover',$ $'student\_id'(Something0).$

Table 2. A student's policy, in CNL and its corresponding Protune translation.

Since policies are declarative statements and by using a current state-of-the-art policy engine, we allow advanced control of data sharing. We will now shortly name those features [19]—instantiated as the features of Protune—and describe their benefits for social software:

Negotiations: In traditional access control or authorization scenarios only one party is able to specify policies which the other one has to conform to. Typically only one of the interacting parties is enabled to specify the requirements the other has to fulfill, whereas the other has no other choice but satisfying them (and thereby being authorized) or not (and thereby not being authorized). Therefore, a more expressive approach allows both parties to discuss (i.e., negotiate) in order to reach an agreement. Trust Negotiations [3], for example, allow both parties to incrementally disclose information in order to get or grant access to some data on a social platform. As an example, see the policies in the Tables 1 and 2: if the student requests access to a lecturers document that is protected, she is asked to provide her student id. But the student herself protects her student id by Policy P3 stating that any party that requests the student id has to be an employee of the University of Hannover. Consequently, the lecturers request for a student id leads to the student's request for the lecturer's employee credential. Since this credential is not protected (see Policy P2), it is disclosed, therefore the student discloses the student id and is allowed to access the protected document. Mind that, first, this procedure is automatically performed by the parties' Protune clients and that, second, without such a negotiation, the student would not have got access to the document.

Evaluation and Actions: In order to evaluate a policy, some actions performed by the policy infrastructure might be required. Examples for such actions are sending credentials or queries to external systems, for example in order to decide if the requester is a registered friend on some other social platform. In this case, the Protune policy enforcement infrastructure can automatically query this platform to get the required information. Another common action is the retrieval of environmental properties like the current system time (e.g., if access is allowed only in a specific time frame) or location (e.g., share files only with people in the same meeting room). By doing this, privacy control in social platforms becomes extensible and information from other accessible social platforms can be exploited for security decisions.

Explanations: As stated in Section 3.1, with Protune it is possible to generate explanations [20] out of the policies and the decisions they make. On the one hand, this helps a user to check whether the policies she created are correct and, on the other hand, they inform other users about why a decision was taken (or how the users can change the decision by performing a specific action). For example, whenever access to a certain data or resource is denied on a data sharing platform, an explanation why the request failed can be provided. Given the example from Tables 1 and 2, a requester that does not disclose her student id would get an explanation such as "I cannot find a Credential such that Credential is a student id." Together with the authoring of policies in CNL, such natural language explanations encapsulate the formal policy evaluation completely from the user.

Strong/lightweight evidences: The result of a policy's evaluation may depend on the identity or other properties of a requester such as age, membership in a certain club, etc. Protune provide a means to communicate such properties. Usually, this is done by sending digital certificates called *strong evidences*. Typical strong evidences are credentials signed by trusted entities called *certification authorities*. *Lightweight evidences* [21] are non signed declarations or statements (e.g., license agreements).

Ontologies: In trust negotiations as they are described above, policies have to be exchanged among entities within the social platform in order to transfer information about what is needed to fulfill a policy. For example, the lecturer's Protune client in the example has to transfer the policy protecting the document in order to let the student's client know that a student id is required. Although the basic constructs may be defined in the policy language (e.g., rule structure and semantics), policies may be used in different applications and even define new concepts. The ontology support of Protune helps to provide well-defined semantics for new concepts to be understood by different entities[22, 23].

All those features of our approach meet the requirements identified in Section 2 as follows. Using the Protune language makes our privacy solution adaptive and fine-grained. Protune's ability to add external data sources allows to extend our solution towards data sources that lay outside of the actual social platform. The natural language policy definition and Protune's explanation facility makes our approach talk natural language to the user and vice versa. Last but not least, our solution comprises the common security features. The logic programming nature of the policy language behind makes the privacy enforcement reliable.

6 Implementation

In this section we present a prototype implementing privacy control in a collaborative platform with natural language policies.[5] We realized the presented approach as an extension to the collaborative learning environment LearnWeb2.0 [24]. LearnWeb2.0 is a Web2.0 application which integrates several Web2.0 tools and allows the user to collaboratively develop new competences while staying in a single comfortable environment. Therefore, LearnWeb2.0 supports sharing and rating of arbitrary resources that help developing competences.

The extension we describe in this paper allows users to exactly define which condition a user has to meet in order to access a resource that is shared on the platform. These policies can be defined in controlled natural language policies and are subsequently translated into Protune policies and enforced whenever a user requests access to a resource belonging to a

[5] The prototype is accessible at http://learnweb.L3S.uni-hannover.de.

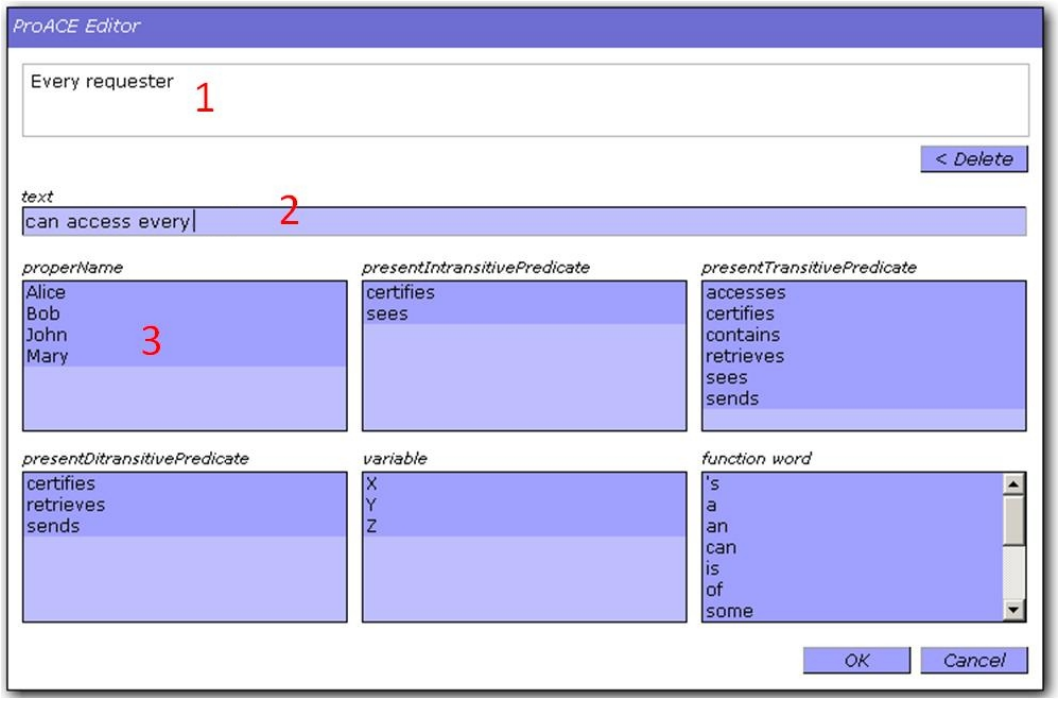

Fig. 1. Creating a policy with the ACE policy editor.

collaborator. Before we detail our extension, we first shortly introduce the features of the LearnWeb2.0 platform.

6.1 LearnWeb2.0

The main goal of LearnWeb2.0 is to help the user to organize, rate, and manage resources that support the development of her desired competences. In order to create one's own specific set of learning resources, there are three ways to add resources to the repository. First, the user search via the LearnWeb2.0 search utility for publicly available content in a collection of Web2.0 tools such as YouTube, Flickr, Ipernity, GroupMe!, and Delicious. The user can then select resources that correspond to the desired competence and add them to her LearnWeb2.0 repository. Second, LearnWeb2.0 offers the user a virtual desktop facility with access to all resources (including private ones) stored on different Web2.0 applications in her own accounts. Third, resources can be directly uploaded from the user's desktop. Therefore LearnWeb2.0 offers a common upload interface which stores the uploaded resources on suitable Web2.0 accounts that are connected to LearnWeb2.0. Given this upload feature, a user can store pictures on Flickr, videos on YouTube, audio tracks on Ipernity and the like from one single platform, namely LearnWeb2.0. Furthermore, resources can be commented, tagged and rated by other LearnWeb2.0 users who are authorized to access them according to the resource owner's policy. Thus a group of users can be established that share learning material and therefore are supported by collaboratively developing a particular competence.

6.2 The ACE Policy Editor

Fig. 1 is a screenshot of the ACE Policy Editor a user can use in order to define Protune policies by means of ACE sentences. This editor is based on the predictive authoring tool[6] developed by the ACE group. The ACE Policy Editor guides the user step by step during the creation or modification of policies and makes sure that only sentences are created which can be translated into Protune policies. Therefore the translation of such sentences into Protune policies will never lead to errors.

The text field marked with (1) is read-only and shows the beginning of an ACE policy. This field is automatically updated whenever the user deletes the last inserted token (by pressing the button "Delete") or add a new token which has been accepted by the editor (cf. below). The text field (2) can be used for entering the next words of the sentence. Whenever the "OK" button is pressed, such words are moved to the text field (1) up to the point where a word is not a correct continuation of the already entered sentence. From this point on, the remaining of the text is kept in (2). Clicking on the entries of the lists (3) is an alternative way to construct a sentence. Each and all words which can be entered at the current position of the sentence are shown according to their grammatical category. In the depicted case, only proper names, intransitive predicates, and so on are allowed. The list for common nouns, for example, is not shown because common nouns are not allowed to appear behind the common noun *requester* in the depicted sentence.

6.3 Extending LearnWeb2.0 with Natural Language Privacy Policies

We extended LearnWeb2.0 by a policy management dialog (see Figure 2). For each policy the title and the text in natural language is returned and displayed in the table. The user has a possibility to edit each of them, or add a new one by supplying a policy title to the creation form and clicking on the submit button. For creating and editing policies LearnWeb2.0 calls the ACE policy editor described in the previous section. The created policies are first translated to Protune policies and then submitted to a Protune server responsible for the policy evaluation.

Policy enforcement is integrated by quering the Protune server whenever a user requests access to another user's resource. For example, in the search function of LearnWeb2.0, only those resources are accessible for the user where the policy evaluation succeeded. Currently, the result of a policy evaluation from Protune server we are using is restricted to boolean decisions. That is, either a policy request fails or does not. Therefore, currently, trust negotiations are not possible since they require more advanced evaluation results describing the required steps that make the negotiation succeed.

7 Conclusions and Further Work

In this paper we describe a flexible solution for access control in social platforms based on controlled natural language policies. This approach allows users to exactly define with whom they want to share their data whereas current solutions require the user to fill out predefined forms with only a subset of the possible privacy conditions. Moreover, we allow the user to

[6] http://attempto.ifi.uzh.ch/aceeditor/

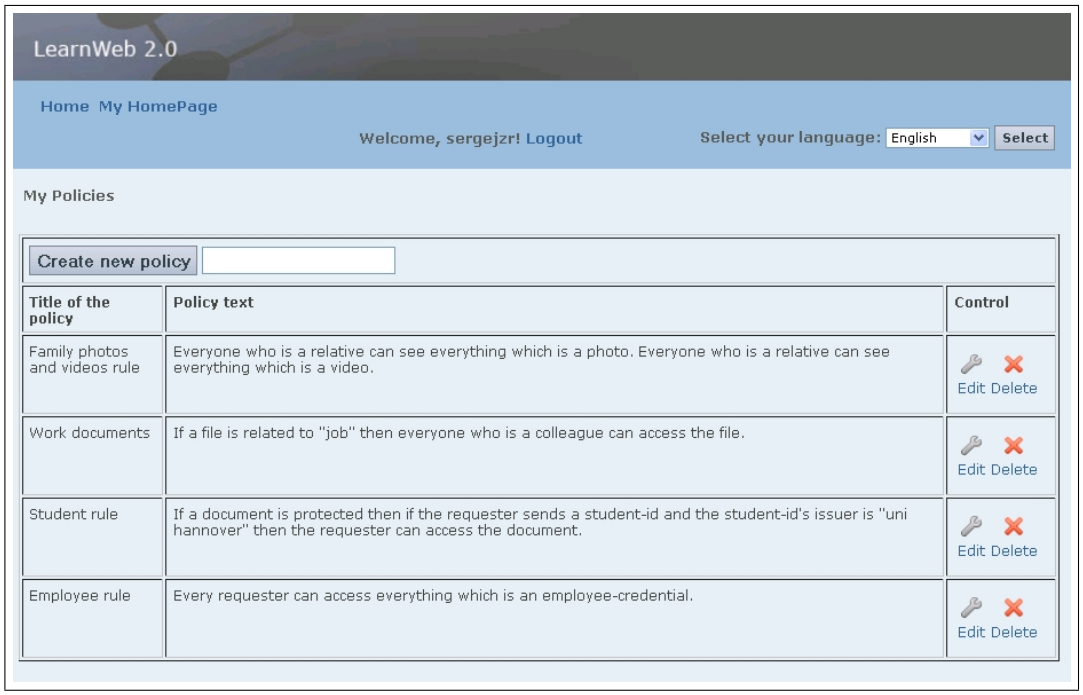

Fig. 2. The LearnWeb privacy settings featuring natural language policies.

define such policies in controlled natural language. We implemented this approach as an extension to the collaborative environment LearnWeb2.0 based on the policy framework Protune and the controlled natural language ACE.

As future work, we plan to extend our implementation to serve for all feature provided by Protune such as explanations and negotiations. The translation of natural language policies to Protune policies still requires careful extensions toward meta-policies. Finally, extensive user studies are on our agenda.

References

1. Passant, A., Kärger, P., Hausenblas, M., Olmedilla, D., Polleres, A., Decker, S.: Enabling trust and privacy on the social web. In: W3C Workshop on the Future of Social Networking, Barcelona, Spain (January 2009)
2. Gross, R., Acquisti, A., Heinz, III, H.J.: Information revelation and privacy in online social networks. In: WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, New York, NY, USA, ACM (2005) 71–80
3. Winslett, M.: An introduction to trust negotiation. In: iTrust. (2003) 275–283
4. Nakashima, E.: Feeling betrayed: Facebook users force site to honor their privacy. The Washington Post (November 30, 2007)
5. AP with Asher Moses: Virgin sued for using teen's photo. The Sydney Morning Herold (September 21, 2007)

6. Chew, M., Balfanz, D., Laurie, B.: (under)mining privacy in social networks. In: Web 2.0 Security and Privacy (in conjunction with IEEE Symp.on Security and Privacy). (2008)
7. Bonatti, P.A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies - a discussion of requirements and research issues. In: ESWC. Volume 4011 of Lecture Notes in Computer Science., Budva, Montenegro, Springer (2006)
8. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: Semantic web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In: International Semantic Web Conference. (2003) 419–437
9. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. POLICY 2003 (2003) 93
10. Kagal, L., Finin, T.W., Joshi, A.: A policy based approach to security for the semantic web. In: ISWC 2003, Springer (2003)
11. Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K.E., Winslett, M.: No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In: ESWS 2004, Heraklion, Crete, Greece, Springer
12. Bonatti, P.A., Olmedilla, D.: Driving and monitoring provisional trust negotiation with metapolicies. In: 6th IEEE Policies for Distributed Systems and Networks (POLICY 2005), Stockholm, Sweden, IEEE Computer Society (June 2005) 14–23
13. Becker, M.Y., Sewell, P.: Cassandra: Distributed access control policies with tunable expressiveness. In: POLICY'04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. (2004) 159–168
14. Coi, J.L.D., Olmedilla, D.: A review of trust management, security and privacy policy languages. In: Int. Conf. on Security and Cryptography (SECRYPT), INSTICC Press (July 2008)
15. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web, Fourth International Summer School 2008. Number 5224 in Lecture Notes in Computer Science, Springer (2008) 104–124
16. Grosof, B.N.: Courteous logic programs: Prioritized conflict handling for rules. Technical report, IBM T.J. Watson Research Center (December 1997)
17. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
18. De Coi, J.L.: Notes for a possible ACE → Protune mapping. Technical report, Forschungszentrum L3S, Appelstr. 9a, 30167 Hannover (D) (July 2008)
19. Coi, J.L.D., Kärger, P., Koesling, A.W., Olmedilla, D.: Control your elearning environment: Exploiting policies in an open infrastructure for lifelong learning. IEEE Transactions on Learning Technologies **1**(1) (2008)
20. Bonatti, P.A., Olmedilla, D., Peer, J.: Advanced policy explanations on the web. In: 17th European Conf. on Artificial Intelligence (ECAI), Italy, IOS Press (2006)
21. Bonatti, P., Samarati, P.: Regulating service access and information release on the web. In: CCS, ACM Press (2000) 134–143
22. Leithead, T., Nejdl, W., Olmedilla, D., Seamons, K.E., Winslett, M., Yu, T., Zhang, C.C.: How to exploit ontologies for trust negotiation. In: ISWC Workshop on Trust, Security, and Reputation on the Semantic Web. CEUR Workshop Proceedings (2004)
23. Nejdl, W., Olmedilla, D., Winslett, M., Zhang, C.C.: Ontology-based policy specification and management. In: ESWC, Heraklion, Greece, Springer (2005)
24. Marenzi, I., Demidova, E., Nejdl, W., Zerr, S.: Social software for lifelong competence development: Challenges and infrastructure. International Journal of Emerging Technologies in Learning (iJET) (2008)